Noname manuscript No.
(will be inserted by the editor)

Multi-step time series prediction intervals using neuroevolution

Paulo Cortez · Pedro José Pereira · Rui Mendes

Received: date / Accepted: date

Abstract Multi-step time series forecasting (TSF) is a crucial element to support tactical decisions (e.g., designing production or marketing plans several months in advance). While most TSF research addresses only single point prediction, prediction intervals (PIs) are useful to reduce uncertainty related with important decision making variables. In this paper, we explore a large set of neural network methods for multi-step TSF and that directly optimize PIs. This includes multi-step adaptations of recently proposed PI methods, such as lower upper bound estimation (LUBET), its ensemble extension (LUBEXT), a multi-objective evolutionary algorithm LUBE (MLUBET) and a two-phase learning multi-objective evolutionary algorithm (M2LUBET). We also explore two new ensemble variants for the evolutionary approaches based on two PI coverage-width split methods (radial slices and clustering), leading to the MLUBEXT, M2LUBEXT, MLUBEXT2 and M2LUBEXT2 methods. A robust comparison was held by considering the rolling window procedure, nine time series from several real-wold domains and with different characteristics, two PI quality measures (coverage error and width) and the Wilcoxon statistic. Overall, the best results were achieved by the M2LUBET neuroevolution method, which requires a reasonable computational effort for time series with a few hundreds of observations.

Paulo Cortez
ALGORITMI Centre, Department of Information Systems, University of Minho, 4804-533 Guimarães, Portugal
Tel.: +351 253510309
E-mail: pcortez@dsi.uminho.pt
ORCID: 0000-0002-7991-2090

Pedro José Pereira
ALGORITMI Centre, Department of Information Systems, University of Minho, 4804-533 Guimarães, Portugal
E-mail: id6927@alunos.uminho.pt

Rui Mendes
Centre of Biological Engineering/ALGORITMI Centre, Department of Informatics, University of Minho, 4710-057 Braga, Portugal
E-mail: rcm@di.uminho.pt
ORCID: 0000-0002-5321-6863

Keywords Prediction intervals · Time series · Multi-objective evolutionary algorithm · Multilayer perceptron · Evolutionary neural network

1 Introduction

Time series forecasting (TSF), which models an event based on its past observations, is a crucial element to support decisions in several real-world domains, such as agriculture, economics, production, commerce and marketing [1]. In particular, multi-step ahead TSF is useful for supporting tactical decisions, such as planning production resources or designing a marketing campaign several months in advance. TSF is usually modeled as a single point prediction task. However, Prediction Intervals (PIs), set in terms of lower and upper bonds for the forecasts, are also invaluable in decision making, allowing to better estimate the uncertainty associated with key decision variables. For instance, a PI can be used to define what-if decision scenarios for the worst and best cases. As explained in [2], a PI includes more sources of uncertainty when compared with statistical confidence intervals, since it includes not only noise but also error measurements, lack of input data and even TSF model misspecification.

Due its importance, there is a vast scientific literature that addresses single point TSF, set in terms of two main approaches: Statistical and Soft Computing methods. Statistical methods are often developed within the field of Operations Research and are based on mathematical expressions [3]. Popular examples include the Holt-Winters method and the ARIMA methodology [1]. Soft Computing approaches are more related with the fields of Computer Science and include a range of distinct methods, such as [4,5]: neural networks (NNs), fuzzy techniques, support vector machines, evolutionary computation and even hybrid combinations of the previous methods. In particular, several works have used evolutionary computation to successfully optimize NN for single point TSF [6,7], in a hybrid combination that is known as neuroevolution [8].

This paper addresses PI TSF using NNs, which is a much less researched topic and that involves two main measures of quality: PI coverage and width [2,9]. The former is set in terms of number of point forecasts included in the PIs, while the latter is defined by upper and lower bond overall difference. These measures often conflict. For instance, reducing a PI width tends to decrease the PI coverage. Thus, a trade-off needs to be set between the two measures. Prior to the year of 2011, several methods were proposed for regression PIs using a NN as the base learner model. For example, a Bayesian method was used in 1992 to assign error bars to the NN predictions [10]. The method requires a high computational effort due to the calculation of derivatives and the Hessian matrix. The delta technique was proposed in 1996 [11]. The technique contains two main limitations, it uses a linear NN model and it assumes that noise is normally distributed. The bootstrap is a more simpler approach that is more suited for small datasets [12,13]. It assumes an ensemble of NNs, each trained on bootstrap replicates of the training set. One limitation of bootstrap is that its adaption to the multi-step forecasting domain is non trivial, since there is a chronological order in the training series elements and boostrapping such elements would lead to sequential information loss. More importantly, as argued in [2], the Bayesian, delta and bootstrap methods contain two methodological drawbacks: the NN model was trained to optimize the single

point prediction and not the PI; and the obtained intervals were only assessed using PI coverage but not width. In contrast, recent works (e.g., LUBE, LUBEX, MLUBE) assume a more natural approach where the PIs are directly optimized when fitting the NN and consider both coverage and with PI quality measures [2, 14, 9, 15]. In this paper, we follow this natural approach, putting a stronger focus on two key state of the art PI models (LUBE and MLUBE) that were proposed for regression and that are extended for multi-step ahead TSF.

In 2011, the lower upper bound estimation (LUBE) method was originally proposed for regression tasks [2]. LUBE uses a multilayer perceptron NN with two output nodes that correspond to the lower and upper PI values. LUBE optimizes a single and nondifferentiable objective function called coverage width-based criterion, which combines both coverage and width. Thus, a simulated annealing metaheuristic was used to train the NN weights instead of the conventional backpropagation algorithm. When compared with traditional PI methods (delta, Bayesian and bootstrap) in regression tasks, the LUBE method achieved competitive results. In 2013, an extension of the LUBE method was proposed for one-step ahead TSF PI prediction [14]. The extension, named LUBEX, included a time lag feature selection and usage of an ensemble of NNs, where the upper and lower values were computed as the average of several individual NN outputs.

The first neuroevolution approach for PIs was proposed in 2013, aiming also at regression tasks and defined by a direct encoding, where the weights of a fixed LUBE NN architecture were optimized using a multi-objective evolutionary algorithm (MOEA) [9]. The proposed MOEA LUBE (MLUBE) achieved interesting results in two tasks related with oil and gas deposition rates, although the model was not compared with LUBE. Nevertheless, MLUBE is more theoretically sound, since it evolves a Pareto front, thus simultaneously improving both coverage and width objectives, while LUBE only optimizes a single coverage/width trade-off. Later on, in 2015 [15], the same MOEA method was also adapted to fit NN to interval-valued time series data. Unlike LUBE, the base NN contained only an output node and the PI was obtained by feeding first the lower input values to the NN and then the upper inputs. Such as in LUBEX, only one-step ahead PIs were considered. The proposed MOEA method compared favorably against an input-valued LUBE for wind speed data, although only a single run was executed and no statistical test was adopted.

In our preliminary work [16], we extended both LUBE and MLUBE for multi-step TSF PI and performed comparative experiments over four times series. The comparison included a more robust evaluation methodology that included the realistic rolling window procedure [17] and the Wilcoxon nonparametic statistical test [18]. In this paper, we present a more comprehensive set of PI TSF methods, putting an emphasis on neuroevolution approaches. The set of PI methods include: adaptations of LUBE, LUBEX and MLUBE for multi-step TSF (LUBET, LUBEXT and MLUBET), a two phase learning MLUBE (M2LUBET) and new ensemble neuroevolution versions (MLUBEXT, M2LUBEXT). We also perform a wider comparative study, using a robust evaluation and nine time series from distinct real-world domains (e.g., Agriculture, Retail, Tourism).

The paper is organized as follows. Section 2.1 presents first the time series data. Then, the forecasting methods are described in Section 2.2. In particular, single-point NN TSF is briefly presented in Section 2.2.1. To better describe all PI adaptations and aiming at self-containment, Section 2.2.2 first details the LUBE original

model and then its LUBET and LUBEXT extensions. Similarly, Section 2.2.3 introduces first MLUBE and then its time series and ensemble variants: MLUBET, M2LUBET, MLUBEXT, M2LUBEXT, MLUBEXT2 and M2LUBEXT2. Next, Section 2.3 explains the evaluation procedure. Section 3 details the performed experiments and obtained results, with both single NN and ensemble NN methods. Finally, Section 4 describes the main conclusions and future work.

2 Materials and methods

2.1 Time series datasets

This paper addresses seasonal series, since multi-step forecasts are particularly relevant for these types of cycle patterns, which are expected to reoccur in the future. Moreover, seasonality is common in several real-world domains (e.g., monthly sales or quarterly production numbers) [1]. Table 1 presents the main characteristics of the nine time series that were selected from distinct domains and with different characteristics. As shown in Figure 1, in addition to the seasonal component the datasets exhibit distinct trend components: some present a growth trend (e.g., gas, pass, water), some are more stationary (e.g, MG, suns) and crad presents a changing trend effect. Seven series (cradfq, gas, pass, pigs, suns, usauto, water) were retrieved from the well known Time Series Data Library (TSDL) public repository [19], while MG is a chaotic series [20] and store was collected and detailed in [21]. Except for MG, all series are from real-world domains. As argued in [6], such real-world datasets are often affected by external phenomena, such as economical cycles or meteorological conditions, making them interesting series that are more difficult to predict.

Table 1 Description of the selected time series datasets (L – series length, K – seasonal period, W – rolling window size, S – rolling window step).

Series	Description (location; years)	L	K	W	S
cradfq	Monthly highest radio broadcasting freq. (Washington, D.C., USA; 1934-1954)	240	12	199	1
gas	Monthly gasoline demand, in gallon millions (Ontario, Canada; 1960 1975)	192	12	151	1
MG	Mackey-Glass synthetic chaotic series (–;–)	783	17	505	9
pass	Monthly airline passengers, in thousands (–; 1949-1960)	144	12	103	1
pigs	Monthly number of pigs slaughtered (Victoria, Australia; 1980-1995)	188	12	147	1
store	Daily number of costumers that entered a sports store (Portugal; 2013)	257	7	221	1
suns	Yearly number of sunspots (–; 1700-1988)	289	10	220	2
usauto	Monthly auto registration numbers, in thousands (USA; 1947-1968)	264	12	193	2
water	Monthly water usage, in ml/day (London, Ontario, Canada; 1966-1988)	276	12	205	2

2.2 Forecasting methods

2.2.1 Single point forecasting

Time series consists of several time ordered values related with the same event: $y_1, y_2, ..., y_L$, where L is the length of the series. An autoregressive forecasting

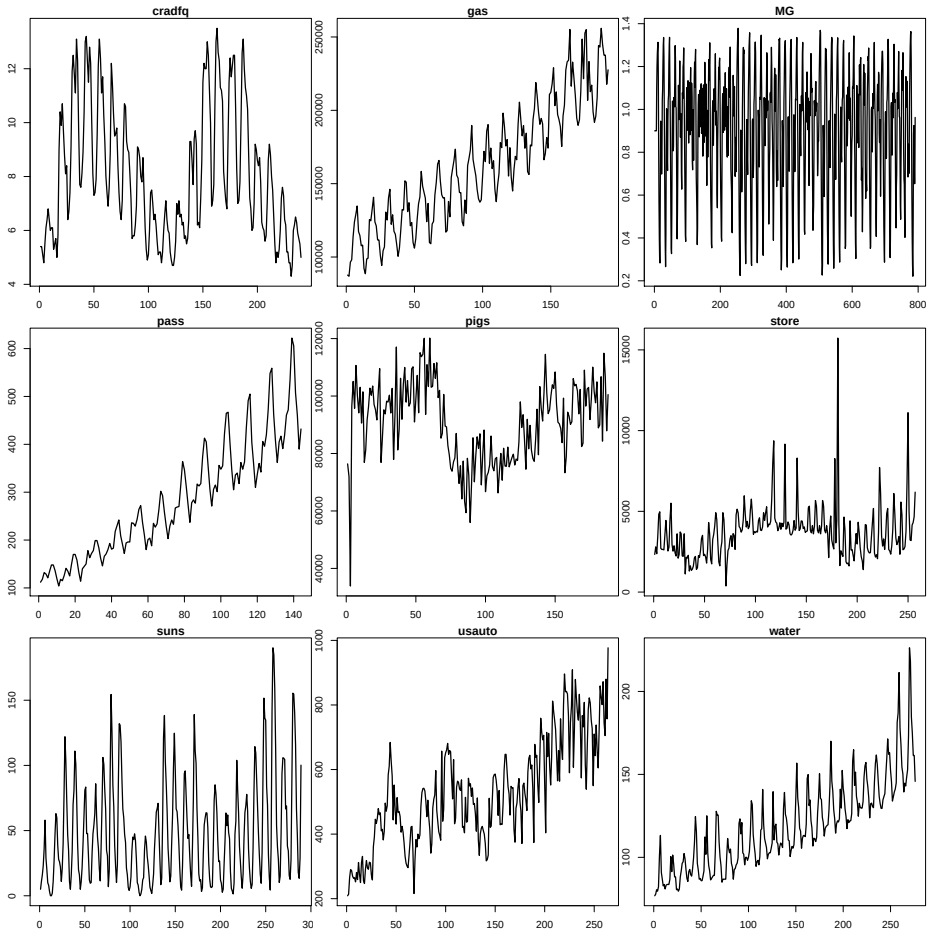

Fig. 1 Time series plots (x-axis denotes the time period t, y-axis the time series y_t value)

model produces the estimate $\hat{y}_{t+1}$:

$$\hat{y}_{t+1} = f(y_{t-k_I+1}, ..., y_{t-k_1+1}) \tag{1}$$

where f is the forecasting function (e.g., linear regression, NN), t is the current time (associated with the last known value y_t) and k_i denotes a time lag. Often, a sliding time window with $\{k_1, ..., k_I\}$ time lags is used to create supervised training cases to fit the learning methods (e.g., multilayer perceptron, support vector machine), thus using a soft computing approach to define the f function. For example, if the $\{1, 3\}$ window is used for the series $5_1, 11_2, 15_3, 17_4, 22_5$ (y_t values) then two training cases can be generated: (5,15) $\rightarrow$ 17 and (11,17) $\rightarrow$ 22.

The fully connected multilayer perceptron is a popular NN for single point TSF [22,4,6]. The regression model is often set in terms of: an input layer with I inputs, the sliding window time lags; a hidden layer with H hidden nodes and logistic activation functions; and an output layer with one output linear function

node. The forecasts are given by:

$$\widehat{y}_{t+1} = w_{2:0,1} + \sum_{j=1}^{H} S(w_{1:0,j} + \sum_{i=1}^{I} y_{t-k_i} w_{1:i,j}) \times w_{2:j,1} \quad (2)$$

where $w_{m:i,j}$ denotes the weight connection with from layer $m-1$ and node i to layer m and node j (if $i = 0$ then it is a bias connection) and S the logistic (sigmoid) function: $S(x) = \frac{1}{1+e^{-x}}$. Figure 2 shows several examples of $w_{m:i,j}$ weight connections.

Ensembles can be used to combine predictions from several forecasting models. Often, ensembles achieve better performances when compared with their individual prediction models [23,24]. In particular, ensemble averaging is a simple and popular approach to combine regression outputs from several learning models [25]. For example, such ensembles have been used to combine single point time series predictions from distinct multilayer perceptrons [22].

To perform multi-step forecasting, several predictions are made at time t, from $\hat{y}_{t+1}$ (1-ahead) to $\hat{y}_{t+h}$ (h-ahead, where h is the forecasting horizon). A popular method to achieve multi-step predictions is to use iterative feedback [6]. Under this method, the model produces first an 1-ahead forecast. Then, this forecast is used as the last input of the forecasting function to generate the 2-ahead prediction, and so on.

2.2.2 Single objective prediction interval methods

This section starts by detailing the LUBE model, which is a fundamental NN structure used by all PI methods explored in this paper. LUBE [2] was the first proposed method to directly optimize PIs. It uses a base learner that is similar to the single point TSF multilayer perceptron (Section 2.2.1), except that the output layer now contains two linear nodes with the lower (L_t) and upper (U_t) prediction interval (PI) bounds, as shown in the left of Figure 2. The number of weights that are optimized is

$$\#\boldsymbol{w}_{\text{LUBE}} = (1+I) \times H + (1+H) \times 2 \quad (3)$$

where the +1 term is due to the use of bias weights ($w_{m:0,j}$), # is the cardinality operator and $\boldsymbol{w}_{\text{M}}$ is the vector that contains all fitted weights ($w_{m:i,j}$) for method M. The NN is trained using a simulated annealing that minimizes the coverage width-based criterion (CWC):

$$\begin{aligned} c_i &= \begin{cases} 1 & \text{if } y_i \in [L_i, U_i] \\ 0 & \text{else} \end{cases} \\ PICP &= \frac{1}{n} \sum_{i=1}^{n} c_i \\ NMPIW &= \frac{1}{R \times n} \sum_{i=1}^{n} U_i - L_i \\ CWC &= NMPIW\left(1 + \gamma(PICP) e^{-\eta(PICP-\mu)}\right) \end{aligned} \quad (4)$$

assuming the prediction interval (PI) of $[L_i, U_i]$ for time period i, where L_i and U_i denote the lower and upper bounds, c_i is the coverage function, n is the number of PI estimates, $PICP$ is the PIs coverage probability, $NMPIW$ is the normalized mean PIs width, R is the range of the target, η and μ are constants, and γ is confidence level step function ($\gamma = 0$ if PICP $\geq \mu$; otherwise $\gamma = 1$). When PICP is higher than the confidence level (μ), CWC produces a small value that is

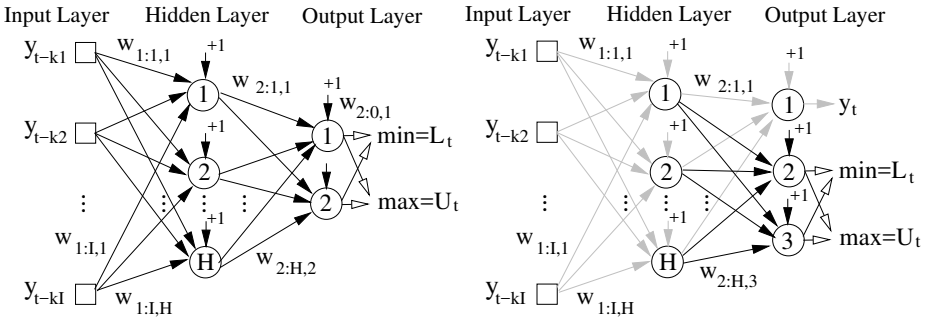

Fig. 2 The base neural network model for LUBE (left) and M2LUBET (right, first phase fixed weights in gray color).

equal to the PI width (NMPIW), otherwise the PI coverage (PICP) is considered unsatisfactory and CWC increases exponentially due to the value of γ.

LUBEX is an extension of LUBE that adopts a time lag feature selection and an ensemble averaging [14]. The latter feature works by first storing the best NR solutions when executing the simulated annealing optimization of LUBE. Then, the PI ensemble is produced by averaging the upper and lower estimates of the distinct NR models.

The μ and η constants are hyperparameters that define how much penalty is given to PIs with a low coverage. In [2], Khosravi et al. assumed the values $\mu = 0.90$ and $\eta = 50$ that highly penalize PIs with a coverage probability lower than 90%. Our claim is that optimizing a single measure, such as CWC, is a more limited approach when compared with the multi-objective neuroevolutionary approaches (Section 2.2.3), since it ignores distinct coverage-width trade-offs. Given that we need to fix *a priori* the μ and η constants for optimizing LUBEX, in this paper we assume the default $\eta = 50$ and $\mu = 0.90$ values proposed in [2]. To achieve a fair comparison, both LUBE and LUBEX are trained with the same inputs (generated by the sliding window with I time lags). To set the best number of hidden nodes (H), the training data is first split into training and validation sets. Then, a grid search is used to explore distinct H values and select the minimum CWC validation network. After setting H, the NN is retrained with all training data.

To facilitate the metaheuristic optimization, we do not assign fixed roles to the two output nodes. Instead, the lower and upper values are automatically set as the minimum and maximum of the output vales. For example, consider a NN with a base structure similar to the left of Figure 2. For a particular time series input window, the NN could return the pair of values (230 – first output node, 137 – second output node). If the NN output node roles were previously fixed to the lower (first output) and upper (second output) then the interval would be infeasible, with $U_t < L_t$. In contrast, the flexible minimum and maximum role assignment leads to a feasible interval: $[\min(230, 137) = 137, \max(230, 137) = 230]$.

LUBE was originally proposed for regression tasks, while LUBEX only handled one-step ahead TSF. In previous work [16], we adapted both methods to perform multi-step ahead TSF PIs. The adaptation, termed here LUBET and LUBEXT, assumes computing the next ahead estimate as the middle of the PI: $\hat{y}_{t+i} =$

$(U_{t+i} + L_{t+i})/2$, where $i \in 1, ..., h$ and L_{t+i} and U_{t+i} represent the predicted lower and upper bounds for time $t + i$. Similarly to the single point multi-step TSF, an iterative feedback of the $\hat{y}_{t+i}$ values is used to generate the i-ahead PIs and middle values.

2.2.3 Neuroevolution prediction interval methods

In this section, the evolutionary multi-objective method (MLUBE) [9] is first presented, since it is the base model for all proposed neuroevolution TSF PI methods. MLUBE was originally proposed for regression tasks and it uses the same fixed LUBE NN architecture and a direct real-valued representation chromosome with a total of $\#\boldsymbol{w}_{\text{MLUBE}} = \#\boldsymbol{w}_{\text{LUBE}}$ (Equation 3) weights. The weights are set by using the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [26] to evolve a Pareto front that contains all nondominated solutions, i.e., the best multi-objective trade-offs. We adopt the NSGA-II algorithm for the multi-objective optimization because it was used in [9,15] and we wish to have a fair experimental comparison, thus using the same algorithm to compare the distinct MLUBE based methods (MLUBET, M2LUBET, MLUBEXT and M2LUBEXT).

The NSGA-II algorithm optimizes a population with a size of P_s individuals that are initially set randomly, where each individual (or solution) is composed of a vector of real values (the NN weights). In each new generation, new individuals are created by using two genetic operators [26]: a single-point crossover, with a crossover probability of Pc; and polynomial mutation, with a mutation probability of P_m. Then, all individuals are evaluated, using two fitness functions: $PICE = 1 - PICP$ (the PIs coverage error) and $NMPIW$ (the PIs width). The NSGA-II algorithm selects for the next generation the best fitted individuals by using a non-dominated Pareto sorting approach and that considers both fitness functions. This allows the algorithm to evolve iteratively a Pareto front, until it is stopped after G generations.

The LUBE method uses an internal validation procedure to select the best number of hidden nodes (H) based on the single objective CWC metric. MLUBE also uses a similar H value procedure but with an adapted selection criterion. Since MLUBE simultaneously evolves two objectives ($PICE$ and $NMPIW$), instead of CWC we use the highest hypervolume [27] selection criterion, which correlates with interesting Pareto fronts. In effect, the hypervolume is a popular measure used to compare Pareto fronts generated from distinct optimizations and it is computed by considering a reference point. Figure 3 shows an example Pareto front. Often, the reference point is set as the worst solution. In this work, we assume such worst point reference as 100% PICE and 100% NMPIW, i.e., $(1, 1)$. Thus, the ideal hypervolume value is 1.0. Since MLUBE was originally only proposed for regression tasks, we also adapted this method for multi-step PIs (denoted by the term MLUBET), where the middle of the PI is used to provide iterative feedback values ($\hat{y}_{t+i}$) [16].

In our previous work [16], we introduced two variants for MLUBET, both based on a 2-phase learning (M2LUBET). The first phase involves a backpropagation training of a single point TSF multilayer perceptron, with one output node (Section 2.2.1). Then, the obtained weights are fixed (gray connections shown at the right of Figure 2). In the second phase, two additional output nodes, and their

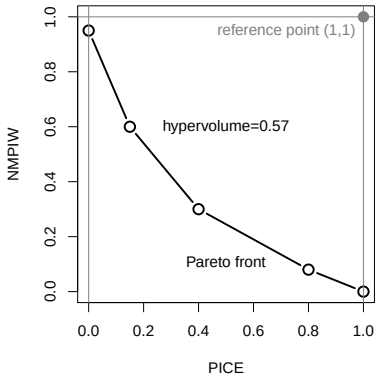

Fig. 3 Example of a Pareto front with five PI coverage and width tradeoffs (white circles) and its hypervolume value when using the (1,1) reference point.

respective connection weights, are added to the model, in order to allow the estimation of the lower and upper values. Next, the new second phase weights ($w_{2:i,j}, j \in \{2,3\}$) are optimized using NSGAII (black connection weights shown at the right of Figure 2). Thus, the multi-objective optimization is performed over a smaller search space when compared with MLUBET, since $\#\boldsymbol{w}_{\text{M2LUBET}} = (1+H) \times 2$.

The difference between the two variants (M2LUBET1 and M2LUBET2) is related with the generation of multi-step PIs. M2LUBET1 uses the first output node to directly generate the $\hat{y}_{t+i}$ estimates, while M2LUBET2 uses the middle PI point (average of the second and third outputs). In this paper, we only consider M2LUBET2 as the M2LUBET representative, since it adopts the same multi-step approach used by other methods (e.g., LUBET, MLUBET) and it provided better results in the preliminary experiments conducted in [16].

The ensemble averaging adopted by LUBEX provided better PI results in terms of the single CWC criterion [14]. Inspired in this result, we propose new ensemble versions for neuroevolution PI methods. These ensemble adaptations are more complex than LUBEX, since multi-objective methods optimize a Pareto curve, with distinct coverage-width trade-offs. Thus, KC ensembles need to be built, each one related with a distinct optimization region. To achieve this, we first store all Pareto curve solutions achieved during the NSGA-II evolution. Then, we split the optimized coverage-width space into KC zones according to two main strategies: radial slices and clustering. The former strategy considers all evolved Pareto curve points, dividing them into KC equal sized radial slices (left of Figure 4). The latter approach clusters the most recent Pareto curve points (from the last LG generations, which tend to include points closer to the best Pareto front) into KC groups using the k-means algorithm [28] (right of Figure 4). Next, we select the best set of NR multilayer perceptrons for each of the KC zones. In both strategies, the selected NR models are the ones that are farther away from the (1,1) reference point, using the Euclidean distance, in coverage-width space. Finally, the PIs of these NR models are then averaged (as in LUBEX), in order to compute the ensemble PI. In this paper, the radial slice ensembles for MLUBET and M2LUBET are termed MLUBEXT and M2LUBEXT, while the clustering ensembles are denoted as MLUBEXT2 and M2LUBEXT2.

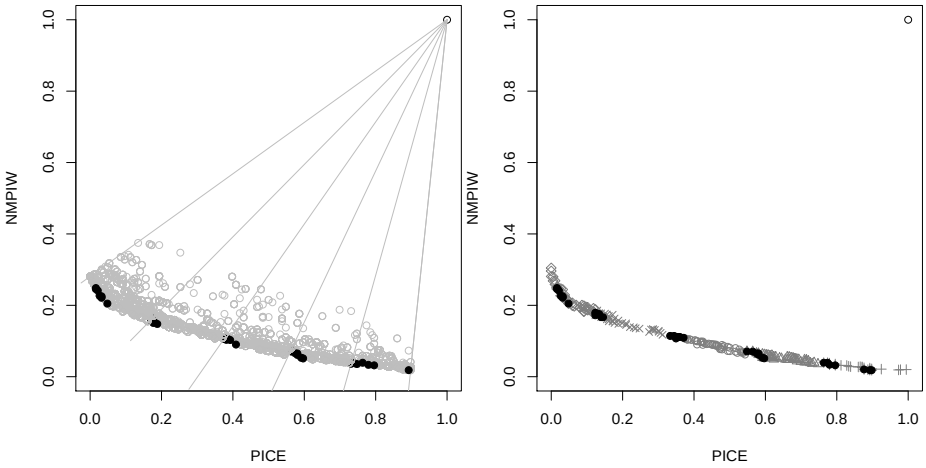

Fig. 4 Example of the radial slices (left) and clustering (right) strategies for ensemble model selection ($KC = 5$ and $NR = 7$; the analyzed Pareto points are in gray; the left graph ensemble regions are separated by radial slices, while each right graph cluster is denoted by a distinct symbol; the selected NR models are in black).

2.3 Evaluation

Forecasting methods are often compared using a time ordered holdout, where the data is split into training and test elements [22]. In this work, we adopt the more robust and realistic rolling window estimation [17], which allows the generation of several training and testing iterations. The rolling window contains three main parameters (Figure 5): W – the size of the training window; h – the number of multiple step ahead predictions; and S – the step size, i.e., the number of window elements are updated in each new iteration. In the first iteration, the training set is composed of the oldest W elements of the time series and it is used to fit the PI method, which estimates, at time $t = W$, 1 to h multistep ahead PIs. Then, the PICE and NMPIW forecasting measures are computed using the time series test samples from time $t+1$ to $t+h$. In the next iteration, the training window is updated by by deleting its oldest S values and adding the more recent $t+S$ time series observations. The forecasting model is then retrained and at time $t = W+S$ new h multi-step ahead forecasts and performance measures are computed, and so on. In total, the rolling window produces $U = \big(L-(W+h-1)\big)/S$ iterations.

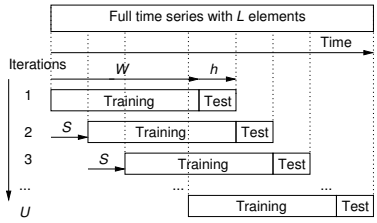

Fig. 5 Schematic of the rolling window procedure.

In order to achieve a more fair comparison, the same inputs are used for all tested PI methods, assuming a sliding time window with all time lags up to $I = K+1$ ($k_i \in \{1, 2, ..., K, K+1\}$), which allows the inclusion of the seasonal pattern (K) plus a possible trend [4]. Table 1 presents the K values for the selected time series.

The number of multilayer perceptron hidden nodes (H) is fixed for each PI base model (LUBET, MLUBET or M2LUBET). To reduce the computational effort, H is only optimized during a preprocessing stage that is performed before the rolling window execution. Using the first training data (oldest W time series observations), the optimization uses a grid search where several H values are tested by splitting the data into fitting (oldest 70% elements) and validation (recent 30% values) sets. Given that the multilayer perceptron training is stochastic, a total of R runs are executed for each H value. Then, the H value is selected, by considering the best average CWC (LUBET, LUBEXT) or hypervolume (other methods) validation measures, and the normal rolling window procedure is executed.

To compare the PI methods, the test set results need to be aggregated, since there are U test sets. For the multi-objective methods, this results in U distinct Pareto fronts. Moreover, some nondominated solutions in the training sets can correspond to dominated points in test sets. As such, the full PI test results often contain several width values for the same coverage, as shown in Figure 6. Thus,

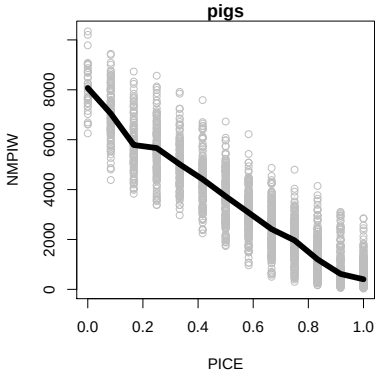

Fig. 6 Example of the forecasting results for MLUBET using the pigs dataset (full test results are in gray; the estimated median curve is in black).

the statistical comparison of PI methods is non trivial. Inspired by receiver operating characteristic (ROC) curve vertical aggregation [29], we propose a similar procedure to compare the PI results from all U iterations. The results are first aggregated vertically, where for each different PICE value the NMPIW median and respective 95% confidence intervals are estimated, according to the nonparametic Wilcoxon test [18]. An example of such median curve is shown in Figure 6. Finally, the overall hypervolume is computed using the estimated Pareto median curve and the adopted (1,1) reference point.

3 Results

3.1 Experimental setup

The experiments were carried out using the R computational environment [30] and run on a dedicated Linux Intel Xeon 1.7 GHZ server. In particular, the multilayer perceptron backpropagation, simulated annealing and NSGA-II fit was performed using the `nnet`, `optim` and `nsga2` functions from the `nnet`, `stats` and `mco` R packages.

In forecasting experimental comparisons, it is quite common to have several modeling parameters. Since its computationally unfeasible to test all parameter combinations, some values need to be fixed using reasonable assumptions [31]. In this work, we adopted the setup that is summarized in Table 2 and that is detailed in the next paragraphs. To compare the methods in a fair way, we used the same evaluation procedure for all PI methods (rolling window parameters, same number of inputs). Furthermore, when possible, we adopted the default algorithm parameters, as suggested in the state of the art works (e.g., [2,14]) or when implemented in the R tool. Similarly to [2], the number of NN hidden nodes (H) is set by using an internal grid search while the ensemble parameters (NR, KC, LG) were set using preliminary experiments.

The R environment default parameter values include: crossover probability of P_c=0.7, mutation probability of P_m=0.2, population size of P_s=100 and maximum of G=100 generations for NSGA-II; and 100 epochs of the BFGS backpropagation algorithm used in the first of the two-phase learning methods. The simulated annealing was set with the values adopted in [2,14] (initial temperature of 5.0 and $\eta = 50$ and $\mu = 0.90$) and it was stopped after $G = 100 \times 100 = 10,000$ iterations, which corresponds to the same level of NSGA-II searched solutions.

Preliminary experiments were held to test several numbers of multilayer perceptrons (e.g., $NR \in \{5, 7, 9, 11\}$) for the ensemble methods and using only the first rolling window training data for series cradfq and MG. Better validation results were achieved for the value $NR = 7$, which was kept fixed for all ensemble methods and time series. In case of the neuroevolution ensemble methods, the coverage-width space was divided into $KC = 25$ regions, which allows to define a large number of trade-offs. We also explored slight different values (e.g., $KC = 23$, $KC = 27$) in preliminary experiments but no substantial differences were achieved and thus we kept the initial $KC = 25$ setup. For the clustering ensemble partition method, LG was set to the last 25 generations.

The training data was first standardized to a zero mean and one standard deviation [32]. Also, all initial multilayer perceptron weights were randomly set within the range $[-1, 1]$. In case of NSGA-II, this means that the lower and upper gene bounds were also set within this range. After training the models, the multilayer perceptron outputs were post-processed with the inverse of the standardized function.

The rolling window evaluation procedure was defined with a reasonable large number of iterations ($U = 30$) and its window and step size parameters (W and S) were adjusted to the time series length, as presented in Table 1. The forecasting horizon was set equal to the seasonal cycle ($h = K$). For instance, the next 12 PIs are estimated for the monthly series, which corresponds to a full year prediction.

Table 2 Summary of the main parameters used in this study

Context	Parameter setup
Time series	seasonal period K is specific to each series (Table 1)
Rolling window	W and S were adjusted (Table 1) to achieve U =30 iterations The horizon h is set equal to the seasonal period K
Inputs	The number of inputs is $I = K + 1$ [4] for LUBE and MLUBE
LUBE	Simulated annealing with initial temperature of 5.0 Simulated annealing stopped after 10,000 iterations Evaluation function of CWC with $\eta = 50$, $\mu = 0.9$ [2,14] Hidden nodes set by internal validation with CWC criterion ($H \in \{0, ..., 9\}$)
MLUBE	NSGA-II with $P_s = 100$, $P_c = 0.7$, $P_m = 0.2$ NSGA-II stopped after G =100 generations Two evaluation functions: $PICE$ and $NMPIW$ Hidden nodes set by internal validation with hypervolume ($H \in \{0, ..., 9\}$)
Ensembles	$NR = 7$ prediction models and $KC = 25$ regions (preliminary experiments) most recent $LG = 25$ Pareto fronts used by the clustering ensembles

Regarding the search of best hidden nodes (H), a total of ten searches were performed using the training data of the first rolling window iteration, in a grid search that ranged from $H = 0$ (simpler linear regression) to $H = 9$ (more complex multilayer perceptron). Each hidden node configuration was trained $R = 10$ times and the average validation estimation measure (CWC or hypervolume) was used to select the best configuration. A grid search was performed for each different base learner, resulting in the three hidden node selections (for LUBET, MLUBET and M2LUBET based models) presented in Table 3. For all series, the single-objective models (LUBET) favor small networks, with just one hidden node. In contrast, larger models are chosen when using the two neuroevolution base models (MLUBET and M2LUBET), ranging from $H = 4$ to $H = 9$.

Table 3 Selected number of hidden nodes (H) for the distinct base learner models

Series	LUBET	MLUBET	M2LUBET
cradfq	1	7	6
gas	1	6	7
MG	1	8	9
pass	1	5	8
pigs	1	6	5
store	1	5	7
suns	1	7	9
usauto	1	7	6
water	1	5	4

3.2 Forecasting methods

Since we compare a large number of PI methods, we first analyze the non-ensemble methods and then the ensemble based ones. Figure 7 shows the non-ensemble

method results in terms of the Wilcoxon estimated median Pareto curve and its respective 95% confidence intervals. The coverage-width forecasting graphs clearly show that LUBET is outperformed by both MLUBET and M2LUBET methods for all time series, even in the low coverage error (PICE) region, which is the area favored by the CWC criterion. In most cases (e.g., cradfq, gas, mg, pass, gas, pass, usauto, water), the neuroevolution results are considerably better, with higher than 0.2 point differences when compared with LUBET in terms of NMPIW for the same PICE value. Moreover, the 95% confidence intervals do not overlap, showing statistically significant differences. Regarding the neuroevolution method comparison, the performances are more similar, presenting smaller differences that depend on the dataset and PICE region analyzed. For instance, M2LUBET shows a substantial improvement over MLUBET for a PICE value around 0.1 for the water series but then a better performance is achieved by MLUBET for PICE values higher than 0.2. For MG and usauto datasets, M2LUBET is consistently better for a large PICE region, while MLUBET outperforms M2LUBET in the same consistent way for pass. The estimated hypervolume values are presented in Table 4, revealing that each method has three best results (cradfq, pass and water for MLUBET; MG, gas and usauto for M2LUBET) and there are three ties. However, the median over all time series favors M2LUBET by 0.06 percentage points.

Table 4 Median hypervolume test values for the neuroevolution methods (best values in **bold**).

	Non-ensemble		Ensemble			
Series	**MLUBET**	**M2LUBET**	**MLUBEXT**	**M2LUBEXT**	**MLUBEXT2**	**M2LUBEXT2**
cradfq	**0.58**	0.56	**0.58**	0.57	0.56	0.56
gas	0.55	**0.57**	0.56	0.47	**0.57**	0.48
MG	0.60	**0.68**	0.60	0.65	0.62	0.62
pass	0.65	0.61	0.63	0.61	**0.66**	0.59
pigs	**0.52**	**0.52**	**0.52**	0.51	**0.52**	0.48
store	**0.73**	**0.73**	0.72	0.72	0.72	0.72
suns	**0.69**	**0.69**	0.68	0.68	0.68	0.67
usauto	0.62	**0.69**	0.59	0.63	0.61	0.67
water	**0.79**	0.74	**0.79**	0.75	0.78	0.75
median	0.62	**0.68**	0.60	0.63	0.62	0.62

Similar results are achieved for the ensemble methods. As shown in Figure 8, the neuroevolution methods have a clear superiority when compared with LUBEXT, presenting high differences in several cases, such as cradfq, gas, pass, usauto and water. Also, the distinct neuroevolution ensembles have closer performances, whose differences depend on the time series and PICE region analyzed. For instance, the two-phase learning ensembles (M2LUBEXT and M2LUBEXT2) are consistently better than other neuroevolution methods for MG data, while the single phase methods (MLUBEXT and MLUBEXT2) are significantly better for the water series. In addition, M2LUBEXT2 provides the best performance for usauto data. The type of region split used for ensemble selection (radial slices versus clustering) does not seem to produce distinctive forecasting behaviors. In several cases,

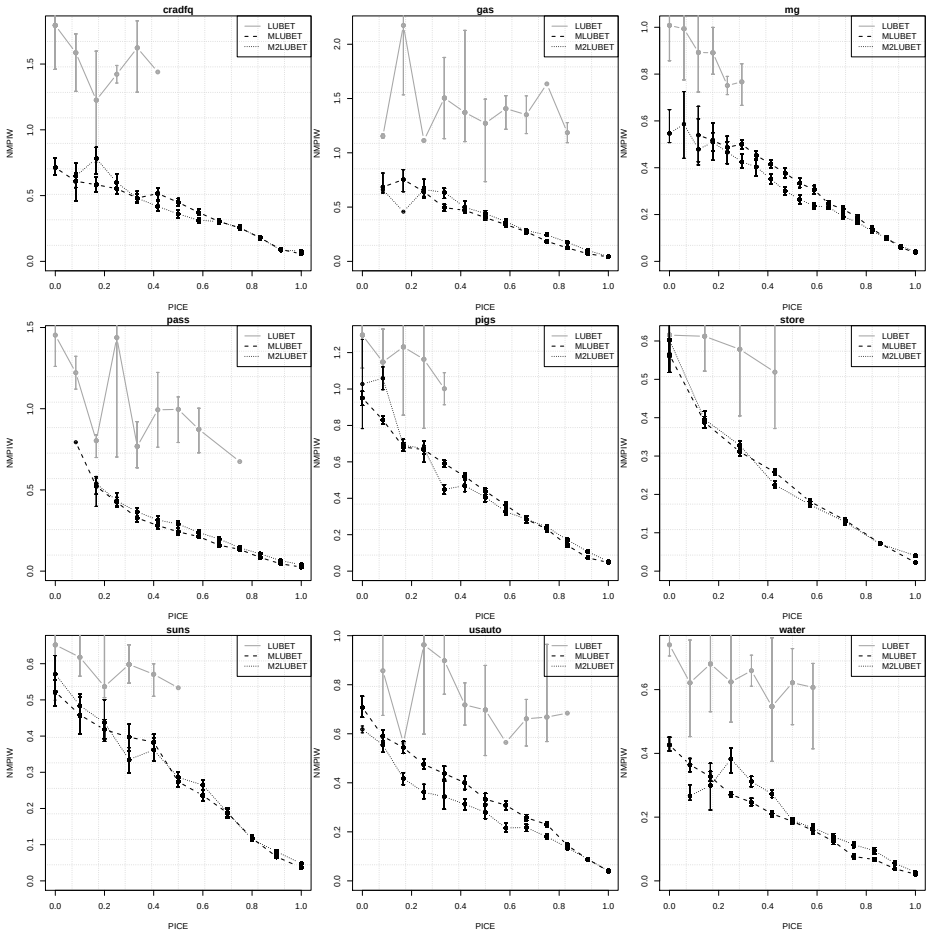

Fig. 7 Forecasting results for non-ensemble methods (points denote the Wilcoxon median values and whiskers represent the respective 95% confidence intervals)

the respective median Pareto curves are aligned. For example, MLUBEXT and MLUBET2 show very similar curves for cradfq, MG, gas, usauto and water. The same similarity effect is visible for M2LUBEXT and M2LUBEXT2 on the series pass and water. The global hypervolume values (Table 4) confirm that there does not seem to be any single superior strategy for ensemble creation. In effect, the best hypervolume performance depends on the dataset analyzed: MLUBEXT – cradfq and water; MLUBEXT2 – gas and pass; and M2LUBEXT – MG; M2LUBEXT2 – usauto. The median values over all series (last row of Table 4) denote small differences, with the overall median values ranging from 0.60 (MLUBEXT) to 0.63 (M2LUBEXT).

When considering the comparison of non-ensemble versus ensemble methods, the results from Table 4 tend to favor the former ones. Indeed, MLUBET or M2LUBET present identical or superior median hypervolume. The only exception occurs with the pass series, where MLUBEXT2 produces the best value. More-

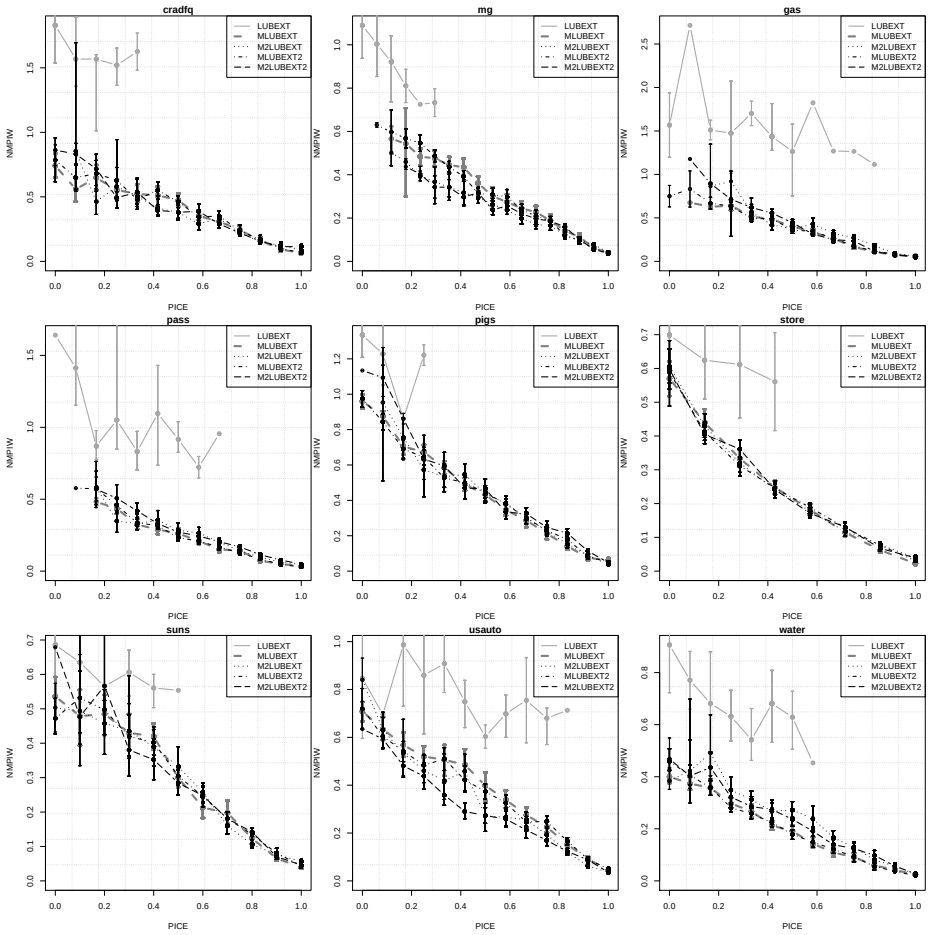

Fig. 8 Forecasting results for ensemble methods (points denote the Wilcoxon median values and whiskers represent the respective 95% confidence intervals).

over, MLUBET is only outperformed by its ensemble variants in three cases (MG, gas, pass), while M2LUBET is only surpassed in two datasets (cradfq and water). In addition, the median values over all series show identical (MLUBET and MLUBEXT2) or superior (MLUBET versus MLUBEXT; M2LUBET versus any of its ensemble variants) performances. When considering all methods and hypervolume values, the best results are obtained by M2LUBET, since it ranks first in six of the nine tested datasets and it also provides the highest overall median value.

To demonstrate the achieved multi-step forecasts, Figure 9 shows examples of twelve multi-step ahead PIs that were computed for two series. The top plots present the MLUBEXT2 predictions for the pass series, while the bottom present the M2LUBET predictions for the usauto data. For each series, two types of forecasts are exemplified (left graphs have smaller PICE values, right graphs have larger ones), allowing to visually compare different coverage-width trade-offs.

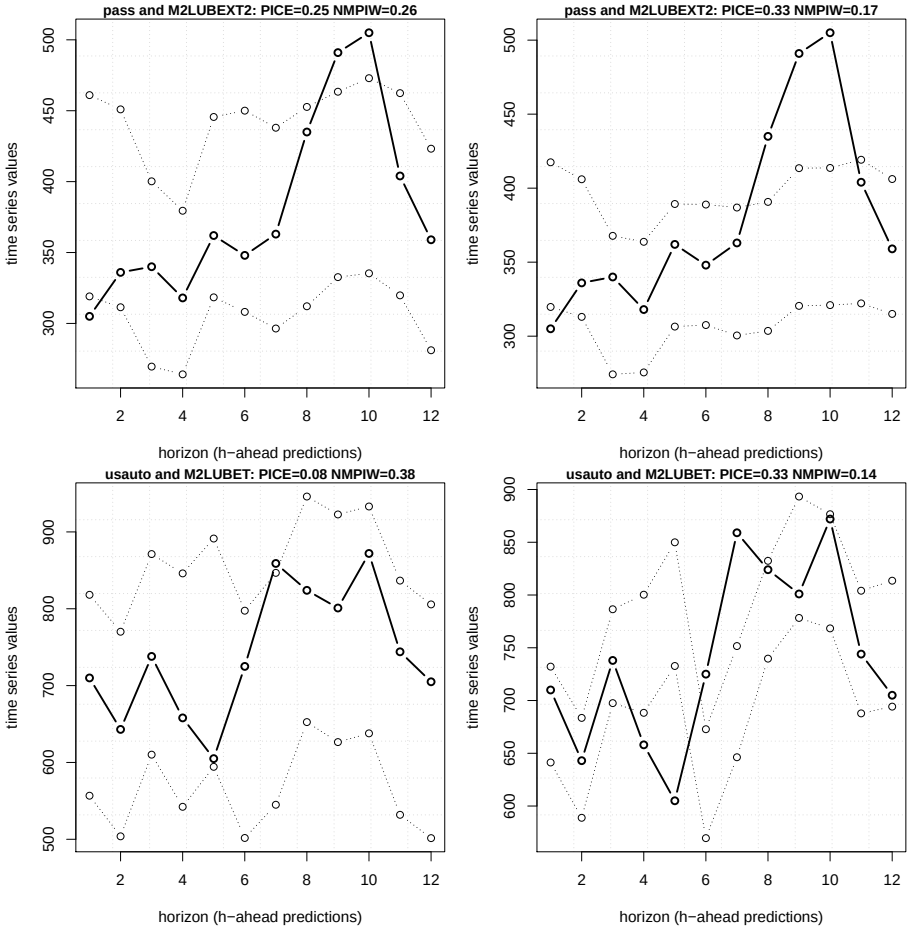

Fig. 9 Examples of the obtained multi-step ahead prediction intervals (solid line denotes the true values; dashed lines represent the upper and lower prediction intervals).

3.3 Computational effort and training optimization examples

The extra computational effort for producing ensembles is negligible when compared with non-ensemble PI method training. Thus, we compare only the computational demand of each PI base learner method and whole rolling window procedure, as detailed in Table 5. The time elapsed values confirm that single-objective methods (LUBET) are faster, requiring around half of the computational effort when compared with the neuroevolution methods. This was an expected result, since LUBET base methods only use one hidden node while other methods use larger H values (Table 3). Furthermore, M2LUBET base methods require around 20% more computational power when compared with the MLUBET ones. More importantly, the computational effort for all PI methods seems highly correlated with the series length, presenting smaller values for the shorter time series (pass) and the highest value for the largest one (MG). Also, it should be noted that the

required effort is reasonable for current computer processors. For instance, a single iteration of the rolling window procedure requires just around 8 minutes (498 s) for the largest series (MG) and most demanding method (M2LUBET).

Table 5 Computational effort for all the base learner methods (in seconds, rows are sorted according to increasing **Lenght** values)

Series		Base Learner Methods		
Name	**Length**	**LUBET**	**MLUBET**	**M2LUBET**
pass	144	1936	3009	3997
pigs	188	2341	4285	5130
gas	192	2379	3977	5306
cradfq	240	2810	5036	6463
store	257	3064	6060	7475
usauto	264	2804	5139	6338
water	276	2863	5376	6721
suns	289	3066	5759	7317
MG	783	5718	11686	14950
median	257	2810	5139	6463

To demonstrate the quality of the neuroevolution optimization, Figure 10 presents examples of the Pareto front convergence for series cradfq and the MLUBET and M2LUBET methods during the first rolling window iteration. In the plots, lines denote the full Pareto front while individual points represent a PI coverage-width trade-off. Also, a gradient coloring was used and that ranges from light gray (first generation) to black (last generation). Both plots reveal a substantial Pareto curve improvement over the optimization. In particular, the training hypervolume increased from 0.54 to 0.83 (MLUBET) and from 0.60 to 0.92 (M2LUBET), when considering the first and last generations of the NSGAII evolution. Moreover, the final Pareto curves are nonlinear and mostly convex, meaning that NSGAII managed to optimize, under a single pass, an interesting range of PI trade-offs that would outperform any linear weighted combination method (e.g., $\alpha \times PICE + (1 - \alpha) \times NMPIW, \alpha \in [0, 1]$).

4 Conclusions

Time series forecasting (TSF) is an important field of research that models past temporal patterns of a phenomenon (e.g., production levels or sales) in order to predict its future values. In particular, multi-step ahead forecasts, computed several time periods in advance (e.g., weeks or months), are useful to support tactical decisions, such as planning production resources. Given the importance of TSF, several single point prediction methods have been proposed, including statistical (e.g., Holt-Winters, ARIMA) and soft computing (e.g., neural networks, support vector machines, fuzzy techniques) approaches. However, there has been much less research that deals with TSF prediction intervals (PIs). And PIs are useful to reduce uncertainty associated with decision making. For example, it can be used to define best and worst what-if scenarios related with strategic decisions.

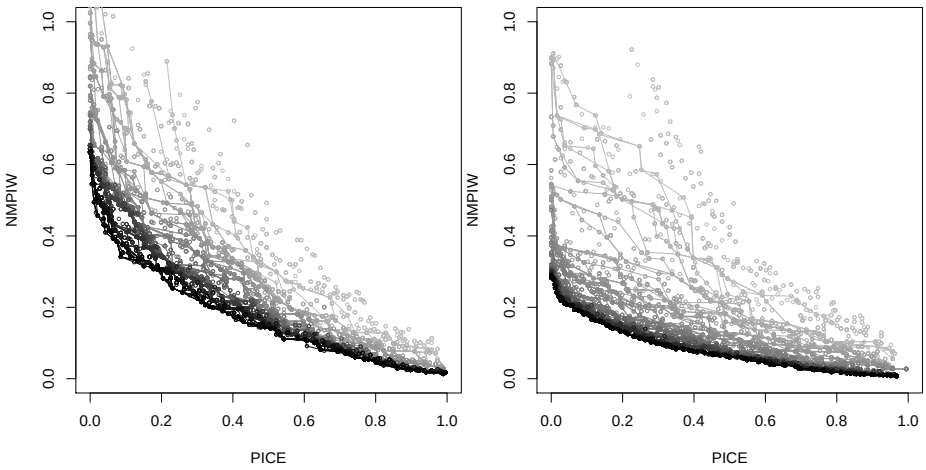

Fig. 10 Evolution of the Pareto front for MLUBET (left) and M2LUBET (right) methods and cradfq series.

This paper addresses this research gap, focusing in soft computing approaches for multi-step TSF PIs. In particular, we adapt and compare a comprehensive set of neural network methods that directly optimize PIs, which includes the lower upper bound estimation method for multi-step TSF (LUBET), the LUBET ensemble extension (LUBEXT) and two neuroevolution methods, namely the multi-objective evolutionary algorithm LUBE (MLUBET) and a two-phase learning MLUBET (M2LUBET). We also present new neuroevolution ensemble variants based on two multi-objective split methods: radial slices (MLUBEXT and M2LUBEXT) and clustering (MLUBEXT2 and M2LUBEXT2).

The PI methods were compared using a robust evaluation that considered a rolling windows procedure, nine time series with distinct characteristics and from distinct real-world domains, two PI criteria (coverage and width) and the Wilcoxon statistic. The obtained results reveal a superior performance of the neuroevolution multi-objective methods when compared with the single objective methods (LUBET and LUBEXT), even when considering the range of models associated with smaller coverage errors. In general, the non-ensemble variants produced similar or slight better results when compared with their ensemble versions. Overall, the best results were achieved by M2LUBET. This method requires a slight computational effort increase when compared with MLUBET. Nevertheless, such effort is still affordable (e.g., requiring around 8 minutes of computation to process a time series with around five hundred elements) and thus we recommend M2LUBET as the best option for multi-step TSF PIs with neural networks.

In future work, we intend to research if better PI ensemble performances can be achieved by exploring other PI aggregation functions, such as usage of an average weighting or stacking [28]. Also, to speedup the computation, we wish to explore multi-core parallel processing (e.g., island models) [33].

Acknowledgements This article is a result of the project NORTE-01-0247-FEDER-017497, supported by Norte Portugal Regional Operational Programme (NORTE 2020), under the

PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF). We would also like to thank the anonymous reviewers for their helpful suggestions.

References

1. S. Makridakis, S. Weelwright, R. Hyndman, *Forecasting: Methods and Applications*, 3rd edn. (John Wiley & Sons, New York, USA, 1998)
2. A. Khosravi, S. Nahavandi, D. Creighton, A.F. Atiya, Lower upper bound estimation method for construction of neural network-based prediction intervals, IEEE Transactions on Neural Networks **22**(3), 337 (2011). DOI 10.1109/TNN.2010.2096824
3. C. Chatfield, *Time-series forecasting* (CRC Press, 2000)
4. M. Stepnicka, P. Cortez, J.P. Donate, L. Stepnicková, Forecasting seasonal time series with computational intelligence: On recent methods and the potential of their combinations, Expert Systems with Applications **40**(6), 1981 (2013)
5. P. Cortez, J.P. Donate, Global and decomposition evolutionary support vector machine approaches for time series forecasting, Neural Computing and Applications **25**(5), 1053 (2014). DOI 10.1007/s00521-014-1593-1. URL `http://dx.doi.org/10.1007/s00521-014-1593-1`
6. J. Peralta Donate, P. Cortez, Evolutionary optimization of sparsely connected and time-lagged neural networks for time series forecasting, Applied Soft Computing **23**, 432 (2014). DOI 10.1016/j.asoc.2014.06.041. URL `http://www.sciencedirect.com/science/article/pii/S1568494614003159`
7. R. Chandra, S. Chand, Evaluation of co-evolutionary neural network architectures for time series prediction with mobile application in finance, Applied Soft Computing **49**, 462 (2016)
8. D. Floreano, P. Dürr, C. Mattiussi, Neuroevolution: from architectures to learning, Evolutionary Intelligence **1**(1), 47 (2008)
9. R. Ak, Y. Li, V. Vitelli, E. Zio, E. López Droguett, C. Magno Couto Jacinto, NSGA-II-trained neural network approach to the estimation of prediction intervals of scale deposition rate in oil & gas equipment, Expert Systems with Applications **40**(4), 1205 (2013). DOI 10.1016/j.eswa.2012.08.018. URL `http://dx.doi.org/10.1016/j.eswa.2012.08.018`
10. D.J.C. MacKay, The evidence framework applied to classification networks, Neural Computation **4**(5), 720 (1992). DOI 10.1162/neco.1992.4.5.720. URL `https://doi.org/10.1162/neco.1992.4.5.720`
11. G. Chryssolouris, M. Lee, A. Ramsey, Confidence interval prediction for neural network models, IEEE Trans. Neural Networks **7**(1), 229 (1996). DOI 10.1109/72.478409. URL `https://doi.org/10.1109/72.478409`
12. T. Heskes, in *Advances in Neural Information Processing Systems 9, NIPS, Denver, CO, USA, December 2-5, 1996*, ed. by M. Mozer, M.I. Jordan, T. Petsche (MIT Press, 1996), pp. 176–182. URL `http://papers.nips.cc/paper/1306-practical-confidence-and-prediction-intervals`
13. R. Dybowski, S.J. Roberts, Confidence intervals and prediction intervals for feed-forward neural networks, Clinical Applications of Artificial Neural Networks pp. 298–326 (2001)
14. M. Rana, I. Koprinska, A. Khosravi, V. Agelidis, Prediction intervals for electricity load forecasting using neural networks, Proceedings of the International Joint Conference on Neural Networks (2013). DOI 10.1109/IJCNN.2013.6706839
15. R. Ak, V. Vitelli, E. Zio, S. Member, An Interval-Valued Neural Network Approach for Uncertainty Quantification in Short-Term Wind Speed Prediction, IEEE Transactions on Neural Networks and Learning Systems (Volume: 26, Issue: 11, Nov. 2015) **26**(11), 2787 (2015)
16. P.J. Pereira, P. Cortez, R. Mendes, in *Progress in Artificial Intelligence - 18th EPIA Conference on Artificial Intelligence, EPIA 2017, Porto, Portugal, September 5-8, 2017, Proceedings*, *Lecture Notes in Computer Science*, vol. 10423, ed. by E.C. Oliveira, J. Gama, Z.A. Vale, H.L. Cardoso (Springer, 2017), *Lecture Notes in Computer Science*, vol. 10423, pp. 561–572. DOI 10.1007/978-3-319-65340-2_46. URL `https://doi.org/10.1007/978-3-319-65340-2_46`
17. L. Tashman, Out-of-sample tests of forecasting accuracy: an analysis and review, International Forecasting Journal **16**(4), 437 (2000)

18. M. Hollander, D.A. Wolfe, E. Chicken, *Nonparametric statistical methods* (John Wiley & Sons, 2013)
19. R. Hyndman, *Time Series Data Library* (http://robjhyndman.com/TSDL/, January, 2010)
20. L. Glass, M. Mackey, Oscillation and chaos in physiological control systems, Science **197**, 287 (1977)
21. P. Cortez, L.M. Matos, P.J. Pereira, N. Santos, D. Duque, in *International Joint Conference SOCO'16-CISIS'16-ICEUTE'16 - San Sebastián, Spain, October, 2016*, vol. 527 (2016), vol. 527, pp. 267–276. DOI 10.1007/978-3-319-47364-2_26. URL `http://dx.doi.org/10.1007/978-3-319-47364-2_26`
22. P. Cortez, M. Rio, M. Rocha, P. Sousa, Multi-scale internet traffic forecasting using neural networks and time series methods, Expert Systems **29**(2), 143 (2012)
23. T.G. Dietterich, et al., Ensemble methods in machine learning, Multiple classifier systems **1857**, 1 (2000)
24. L. Yu, H. Xu, L. Tang, Lssvr ensemble learning with uncertain parameters for crude oil price forecasting, Applied Soft Computing **56**, 692 (2017)
25. N. Oliveira, P. Cortez, N. Areal, The impact of microblogging data for stock market prediction: Using twitter to predict returns, volatility, trading volume and survey sentiment indices, Expert Systems with Applications **73**, 125 (2017)
26. N. Srinivas, K. Deb, Muiltiobjective optimization using nondominated sorting in genetic algorithms, Evolutionary computation **2**(3), 221 (1994)
27. N. Beume, C.M. Fonseca, M. López-Ibáñez, L. Paquete, J. Vahrenhold, On the complexity of computing the hypervolume indicator, IEEE Transactions on Evolutionary Computation **13**(5), 1075 (2009)
28. I.H. Witten, E. Frank, M.A. Hall, C.J. Pal, *Data Mining: Practical machine learning tools and techniques* (Morgan Kaufmann, 2016)
29. T. Fawcett, An introduction to ROC analysis, Pattern Recognition Letters **27**, 861 (2006)
30. R Core Team, *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria (2016). URL `https://www.R-project.org/`
31. D.J. Hand, Classifier technology and the illusion of progress, Statistical science pp. 1–14 (2006)
32. T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd edn. (Springer-Verlag, NY, USA, 2008)
33. D. Sudholt, in *Springer Handbook of Computational Intelligence* (Springer, 2015), pp. 929–959